
git

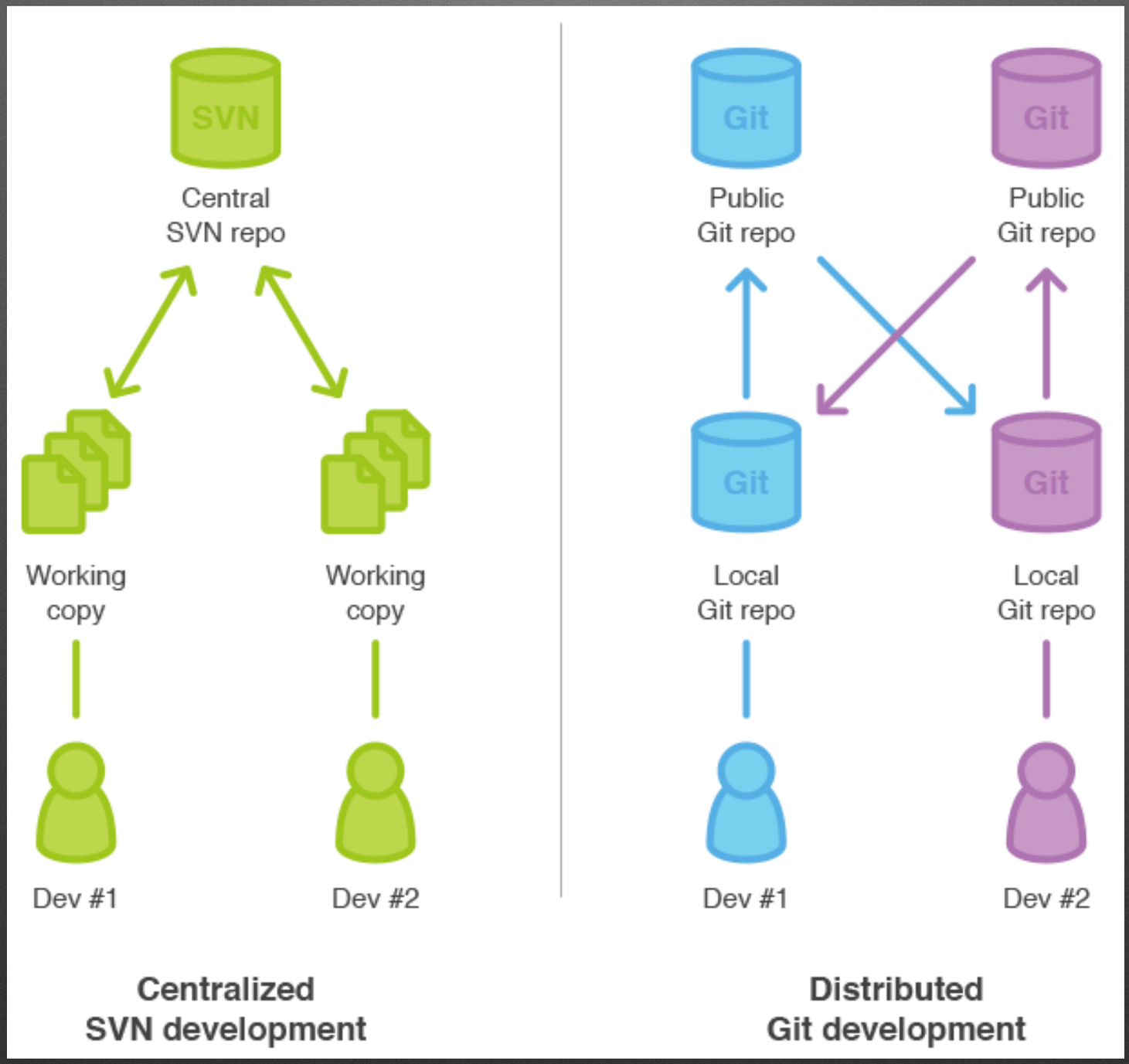
Git

- Who: Git
- What is Git?
- Why should I use Git?
- Where is Git?
- How do I use Git?

What is Git?

- http://www.git-scm.com
  - “Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.”
  - “Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.”

What is Git? - Decentralized



- https://www.atlassian.com/git/migration#!migration-share

Why should I use Git?

- Decentralized
- “cheap local branching”
  - branch for every feature
  - side branches will not affect other users
  - switch context

Where is Git?

- Localhost
  - your computer
- github.com
  - public and private repositories
- private host
  - private server you own

How do I use Git?

Git Cheat Sheet

Remember!
git <COMMAND> --help

Global configuration is stored in ~/.gitconfig.
git config --help

master is the default development branch.
origin is the default upstream repository.

Create

From existing data
```
cd ~/my_project_directory
git init
git add .
```

From existing repository
```
git clone ~/existing_repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://user@host.org/project.git
```

Show

Files changed in working directory
```
git status
```

Changes made to tracked files
```
git diff
```

What changed between ID1 and ID2
```
git diff <ID1> <ID2>
```

History of changes
```
git log
```

History of changes for file with diffs
```
git log -p <FILE> <DIRECTORY>
```

Who changed what and when in a file
```
git blame <FILE>
```

A commit identified by ID
```
git show <ID>
```

A specific file from a specific ID
```
git show <ID>:<FILE>
```

All local branches
```
git branch
```
star (*) marks the current branch

Revert

Return to the last commited state
```
git reset --hard
```
This cannot be undone!

Revert the last commit
```
git revert HEAD
```
Creates a new commit

Revert specific commit
```
git revert <ID>
```
Creates a new commit

Fix the last commit
```
git commit -a --amend
```
(after editing the broken files)

Checkout the ID version of a file
```
git checkout <ID> <FILE>
```

Update

Fetch latest changes from origin
```
git fetch
```
(this does not merge them)

Pull latest changes from origin
```
git pull
```
(does a fetch followed by a merge)

Apply a patch that someone sent you
```
git am -3 patch.mbox
```
In case of conflict, resolve the conflict and
```
git am --resolved
```

Publish

Commit all your local changes
```
git commit -a
```

Prepare a patch for other developers
```
git format-patch origin
```

Push changes to origin
```
git push
```

Make a version or milstone
```
git tag v1.0
```

Branch

Switch to a branch
```
git checkout <BRANCH>
```

Merge BRANCH1 into BRANCH2
```
git checkout <BRANCH2>
git merge <BRANCH1>
```

Create branch BRANCH based on HEAD
```
git branch <BRANCH>
```

Create branche BRANCH based on OTHER and switch to it
```
git checkout -b <BRANCH> <OTHER>
```

Delete branch BRANCH
```
git branche -d <BRANCH>
```

Resolve merge conflicts

View merge conflicts
```
git diff
```

View merge conflicts against base file
```
git diff --base <FILE>
```

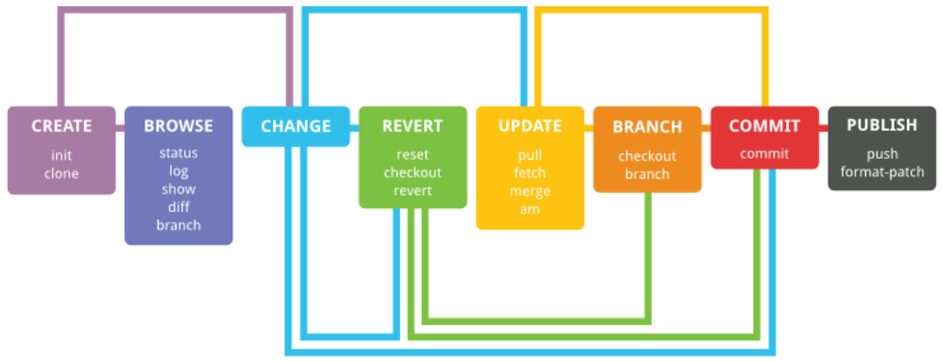
View merge conflicts against your changes
```
git diff --ours <FILE>
```

View merge conflicts against other changes
```
git diff --theirs <FILE>
```

Discard a conflicting patch
```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with
```
git add <CONFLICTING_FILE>
git rebase --continue
```

Workflow



How do I use Git?

- Cheat Sheet: LINK

- Create
- Browse
- (Change)
- Revert
- Update
- Branch
- Commit
- Publish

Use Git: Create

- Create new project
  - `git init`
- Copy existing project
  - `git clone LOCATION`

Use Git: Browse

- Current status information
  - `git status`
- Current changes made
  - `git diff`
  - `git diff FILE_OR_BRANCH`
- Get branch info
  - `git branch`
  - `git branch -av # show all branch info`

Use Git: Update

- Get latest changes from repo
  - `git pull`

Use Git: Branch

- Create branch
  - `git checkout -b BRANCH_NAME`
- Change branch
  - `git checkout BRANCH_NAME`
- Delete branch
  - `git checkout -d BRANCH_NAME`
- Merge branch
  - `git merge BRANCH_NAME`

Use Git: Commit

- Add files to be committed
  - `git add FILE_OR_DIR`
- Commit with message
  - `git commit -m "MY MESSAGE"`
- Commit with message and add all changed files
  - `git commit -am "MY_MESSAGE"`

Use Git: Publish

- Send commits to repository
  - `git push`

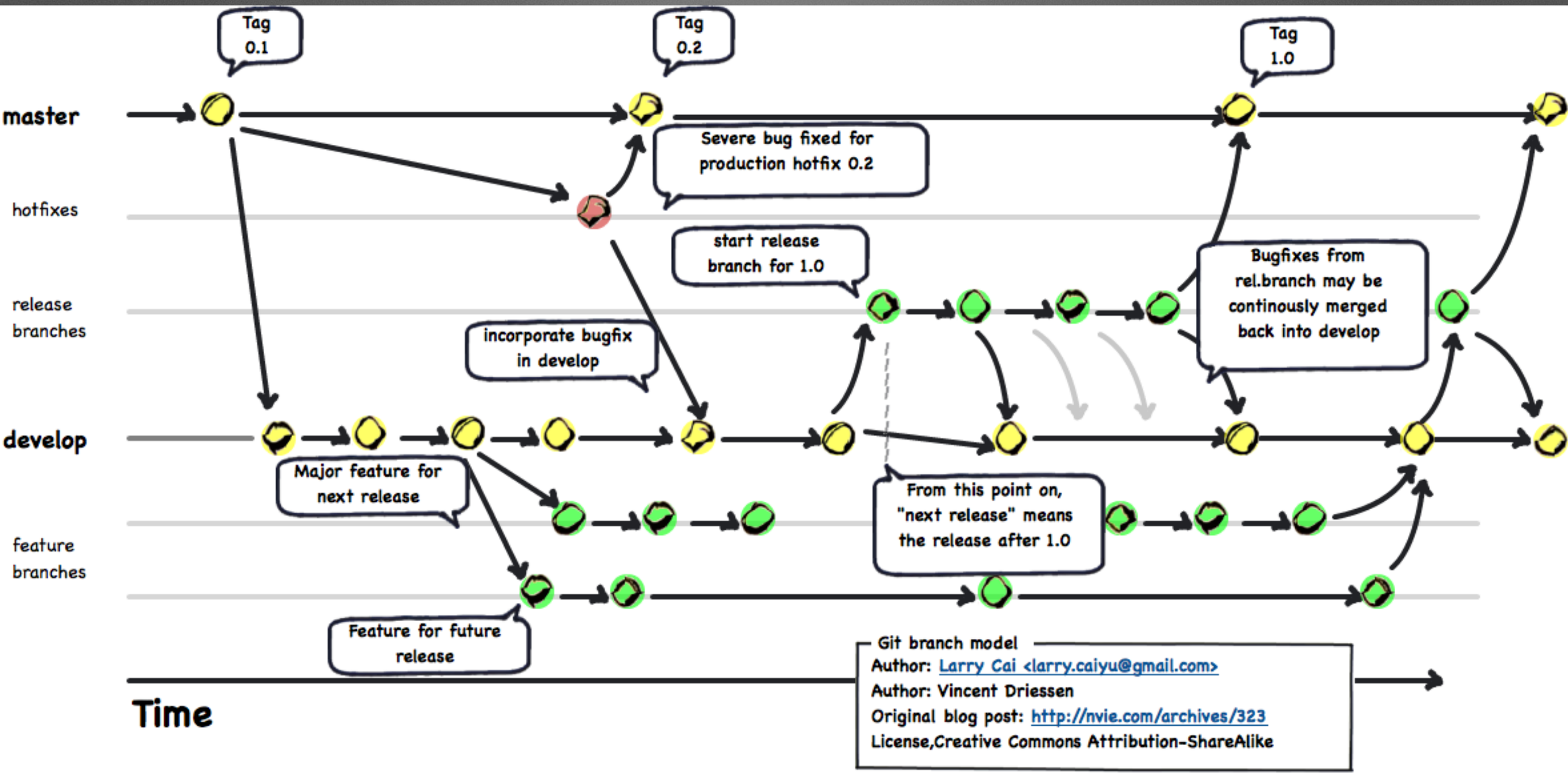
Use Git: Recap

- `git clone LOCATION` `# Copy repository`
- `git status` `# Current status information`
- `git diff FILE_OR_BRANCH` `# Current changes made`
- `git pull` `# Retrieve latest commits`
- `git checkout -b BRANCH_NAME` `# Create Branch`
- `git checkout BRANCH_NAME` `# Change Branch`
- `git checkout -d BRANCH_NAME` `# Delete Branch`
- `git merge BRANCH_NAME` `# Merge Branch`
- `git add FILE_OR_DIR` `# Add changed file`
- `git commit -am "MY_MESSAGE"` `# Commit files with message`
- `git push` `# Publish commits`

Workflow

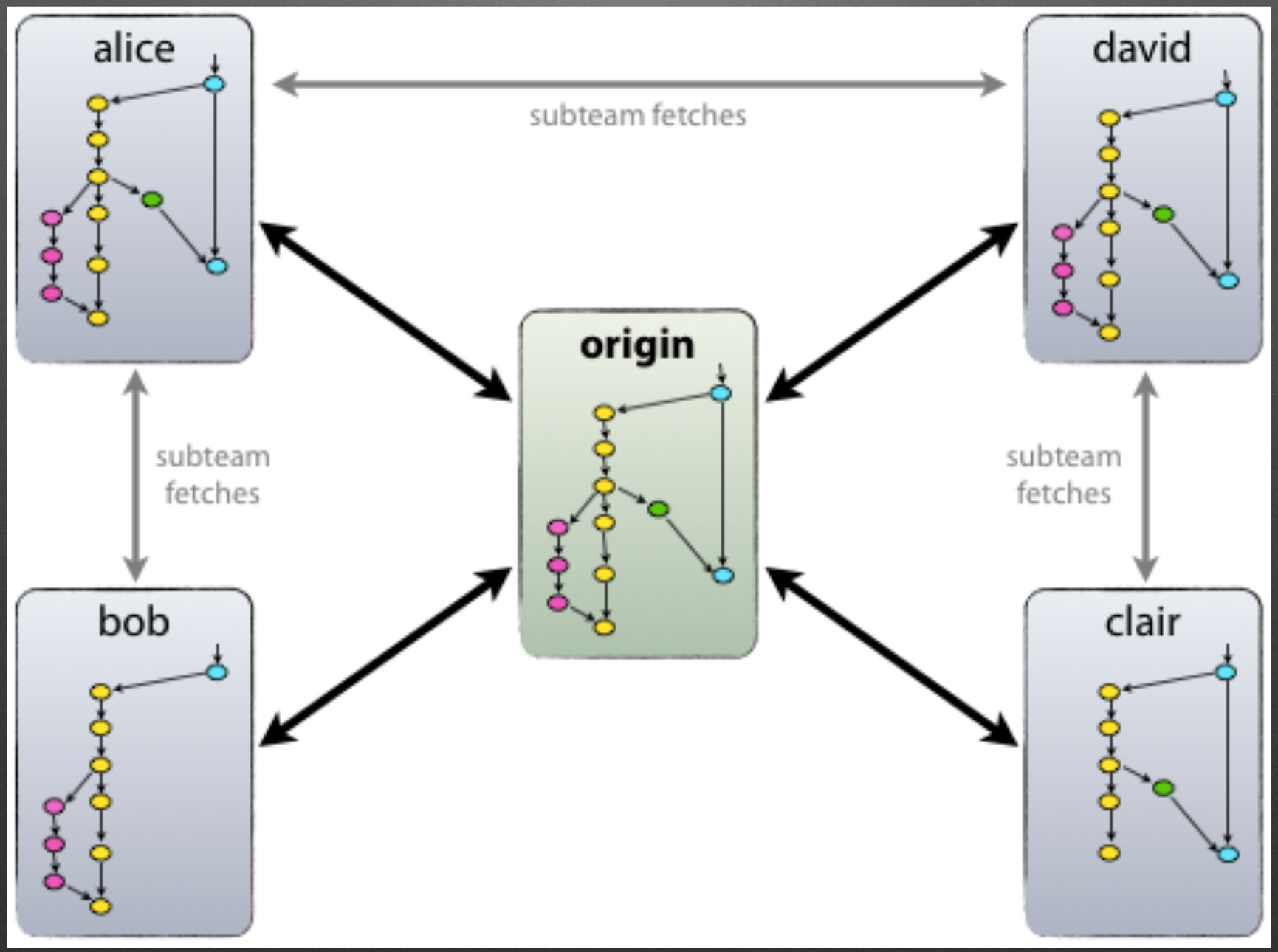
- Successful Git Branching Model Link
  - Read it!!
- The different types of branches we **may** use are:
  - Main branches - should exist forever (master, dev)
  - Feature branches - Make and delete many times
  - Release branches - Backwards compatibility
  - Hotfix branches - Keep to minimum!

Workflow



- https://mockupstogo.mybalsamiq.com/projects/diagrams/Git+Workflow

Decentralized Diagram



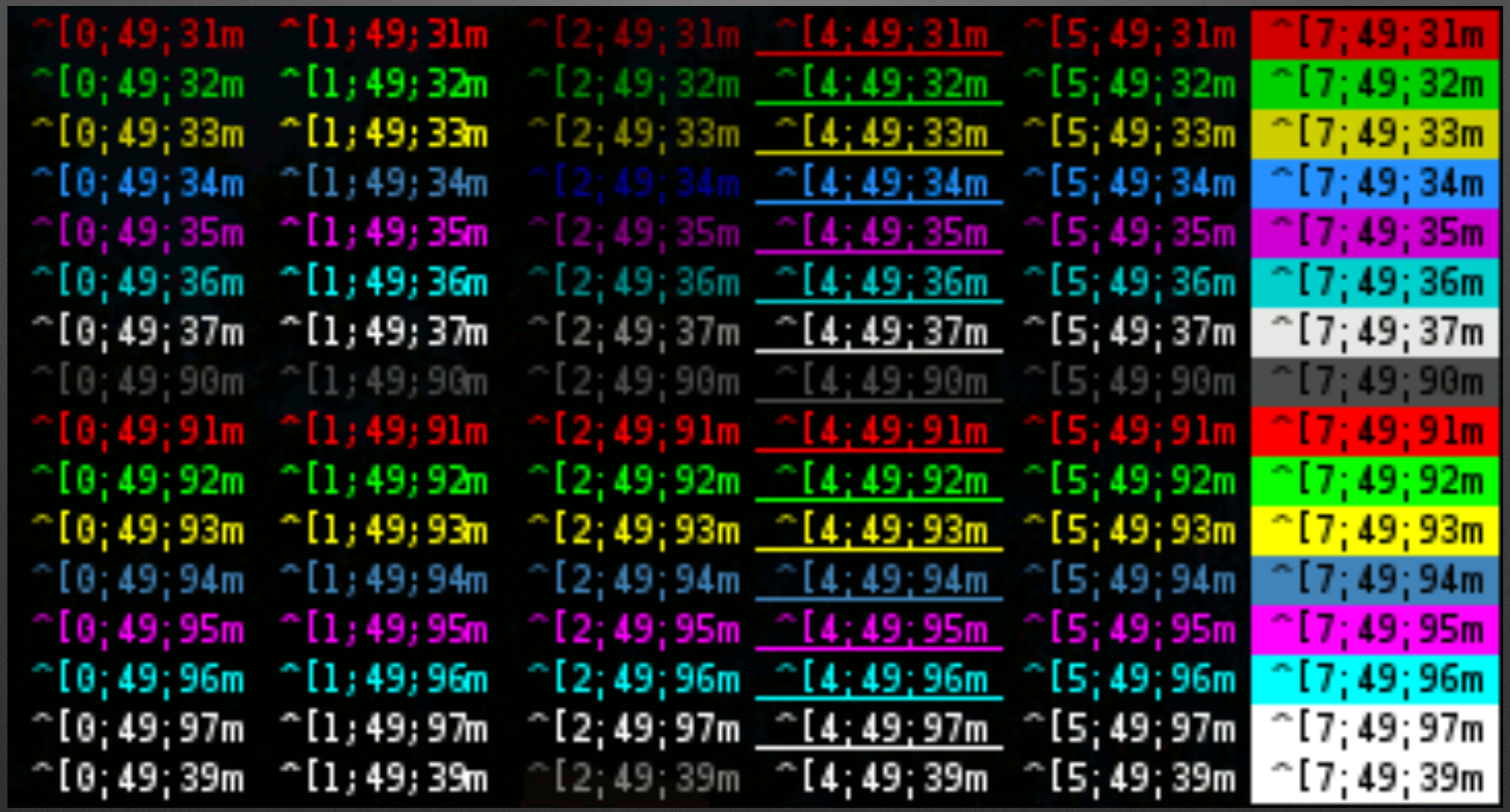
- http://nvie.com/posts/a-successful-git-branching-model/

Github

- https://en.wikipedia.org/wiki/GitHub
  - “GitHub is a web-based hosting service for software development projects that use the Git revision control system.”
    - “GitHub offers both paid plans for private repositories, and free accounts for open source projects. “
- Good for collaboration
- LinkedIn for coding

Git in Command Line

Bash Colors



http://misc.flogisoft.com/bash/tip_colors_and_formatting [7]

Git Color

- File: “~/.gitconfig”
- Will already contain:
  - name
  - email
  - user
  - token

```
[color]
  branch = auto
  diff = auto
  status = auto
[color "branch"]
  current = cyan
  local = yellow
  remote = green
[color "diff"]
  meta = yellow bold
  frag = magenta bold
  old = red bold
  new = green bold
[color "status"]
  added = yellow
  changed = green
  untracked = cyan
```

Example

```
[rCmdCheck] datadr: g b | cat
  dev                          b269661 Update .travis.yml
  master                       b269661 Update .travis.yml
  parallelCheck                6be346a Merge branch 'dev' into parallelCheck
* rCmdCheck                    486b909 [ahead 1] white space
  remotes/origin/HEAD          -> origin/master
  remotes/origin/dev           b269661 Update .travis.yml
  remotes/origin/gh-pages      0280161 update to reflect recent chagnes in package,
  remotes/origin/master        b269661 Update .travis.yml
  remotes/origin/parallelCheck 6be346a Merge branch 'dev' into parallelCheck
  remotes/origin/rCmdCheck     4e29485 updated description license
```

```
[rCmdCheck] datadr: g b
  dev                          b269661 Update .travis.yml
  master                       b269661 Update .travis.yml
  parallelCheck                6be346a Merge branch 'dev' into parallelCheck
* rCmdCheck                    486b909 [ahead 1] white space
  remotes/origin/HEAD          -> origin/master
  remotes/origin/dev           b269661 Update .travis.yml
  remotes/origin/gh-pages      0280161 update to reflect recent chagnes in package,
  remotes/origin/master        b269661 Update .travis.yml
  remotes/origin/parallelCheck 6be346a Merge branch 'dev' into parallelCheck
  remotes/origin/rCmdCheck     4e29485 updated description license
```

Custom Command - g

- github.com/schloerke/g
- Autocomplete functionality given your current github command context

Common Commands - g

- # get current status
  g status
  g st

- # add all files
  g add .
  g a .

- # see latest diff
  g diff
  g d
  g d master

- # see latest logs
  g l

- # checkout other branch
  g co BRANCH_NAME

- # make new branch
  g make_branch BRANCH_NAME

- # undo changes
  g undo FILE FILE2

- # un-commit a file
  g unstage FILE FILE2

Example - g st - git status

```
[rCmdCheck] datadr: g st
  dev           b269661 [origin/dev] Update .travis.yml
  master        b269661 [origin/master] Update .travis.yml
  parallelCheck 6be346a [origin/parallelCheck] Merge branch 'dev' into parallelCheck
* rCmdCheck     486b909 [origin/rCmdCheck: ahead 1] white space
# On branch rCmdCheck
# Your branch is ahead of 'origin/rCmdCheck' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   R/ddo_ddf_kvHDFS.R
#       modified:   R/globals.R
#       modified:   R/mapreduce_kvHDFS.R
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Example - g a <TAB> - git add

```
[rCmdCheck] datadr: g a

add files

# On branch rCmdCheck
# Your branch is ahead of 'origin/rCmdCheck' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   R/ddo_ddf_kvHDFS.R
#       modified:   R/globals.R
#       modified:   R/mapreduce_kvHDFS.R
#
no changes added to commit (use "git add" and/or "git commit -a")

..Rcheck/       .Rbuildignore   .gitignore      DESCRIPTION     NAMESPACE       README.md
.DS_Store       .git/           .travis.yml     LICENSE         R/              data/
```

Example - g l - git logs

```
[rCmdCheck] datadr: g l
* 486b909 - (HEAD, rCmdCheck) white space <Barret Schloerke schloerke@gmail.com> (5 days ago)
* 4e29485 - (origin/rCmdCheck) updated description license <Barret Schloerke schloerke@gmail.com> (5 days
* ffac02b - updated Rbuildignore to have proper grep expression <Barret Schloerke schloerke@gmail.com> (6
*   84bbe5b - Merge branch 'parallelCheck' into rCmdCheck <Barret Schloerke schloerke@gmail.com> (6 days a
|\
| *   6be346a - (origin/parallelCheck, parallelCheck) Merge branch 'dev' into parallelCheck <Barret Schloe
| |\
* | \   3dd830b - Merge branch 'dev' into rCmdCheck <Barret Schloerke schloerke@gmail.com> (6 days ago)
|\ \ \
| | |/
| |/|
| * | b269661 - (origin/master, origin/dev, origin/HEAD, master, dev) Update .travis.yml <hafen rhafen@gma
* | |   54db4d7 - Merge branch 'parallelCheck' into rCmdCheck <Barret Schloerke schloerke@gmail.com> (6 da
|\ \ \
| | |/
| |/|
| * | 3351404 - commented the parallel again <Barret Schloerke schloerke@gmail.com> (6 days ago)
| * | 18194ac - added more files to .Rbuildignore <Barret Schloerke schloerke@gmail.com> (6 days ago)
* | | 915a284 - added documentation for s3method removeData.hdfsConn <Barret Schloerke schloerke@gmail.com
|/ /
* | 7a7e607 - uncommented parallel tests <Barret Schloerke schloerke@gmail.com> (6 days ago)
* | 19f9dfc - white space <Barret Schloerke schloerke@gmail.com> (6 days ago)
* | 0f44726 - renamed LICENSE.md to LICENSE to comply with R CMD check <Barret Schloerke schloerke@gmail.c
* | fa6ad7a - Update .travis.yml <hafen rhafen@gmail.com> (6 days ago)
```

Example - g co <TAB> - git checkout

```
[rCmdCheck] datadr: g co

checkout branch

  dev                          b269661 Update .travis.yml
  master                       b269661 Update .travis.yml
  parallelCheck                6be346a Merge branch 'dev' into paralle
* rCmdCheck                    486b909 [ahead 1] white space
  remotes/origin/HEAD          -> origin/master
  remotes/origin/dev           b269661 Update .travis.yml
  remotes/origin/gh-pages      0280161 update to reflect recent chagne
  remotes/origin/master        b269661 Update .travis.yml
  remotes/origin/parallelCheck 6be346a Merge branch 'dev' into paralle
  remotes/origin/rCmdCheck     4e29485 updated description license

dev             master          parallelCheck
[rCmdCheck] datadr: g co
```

Reduce Working Environment Inefficiencies

- Reduce the amount of “Copy Pasta”
- Set it and forget it
- Minimize keystrokes and thinking required

```
touch “thank_you.txt”

git add ./thank_you.txt

git commit -m “any questions?”

git push
```